Amendments to the Claims:

This listing of claims will replace all prior versions, and listing, of claims in the application:

Listing of the Claims:

1. (Previously Presented) A method of tracking references to objects of an object oriented programming environment, said method comprising:

determining whether a command is likely to place a reference to an object on an execution stack of said object-oriented programming environment;

determining whether there is a change in the flow control when said determining determines that said command is likely to place a reference to an object on an execution stack;

translating said command into another command when said determining determines that there is a change in the flow control; and

placing a reference to said object on a reference stack associated with said execution stack when said another command is executed.

2. (Previously Presented) A method as recited in claim 1, wherein said object-oriented programming environment is a Java compliant operating environment.

3. (Original) A method as recited in claim 2, wherein said determining of whether a command is likely to place a reference on said execution stack is performed during Java Bytecode verification.

4. (Original) A method as recited in claim 3, wherein said determining of whether a command is likely to place a reference on said execution stack operates to determine whether a Getfield, Aload, Getstatic, or Return command is being performed.

5. (Canceled)

6. (Previously Presented) A method as recited in claim 4, wherein said method further comprises:

determining whether a Putfield command is likely to overwrite a reference to an object on the execution stack before said reference is used when said determining determines that there is not a change in the flow control; and

translating said command into another command when said determining determines that there is a Putfield command is likely to overwrite a reference to an object on the execution stack before said reference is used.

7. (Original) A method as recited in claim 1, wherein said reference stack and said execution stack have the same size.

8. (Original) A method as recited in claim 1, wherein at least one reference to an object is stored in an entry with an offset that is the same as the offset used to store said at least one reference in said execution stack, when said another command is executed.

9. (Original) A method of tracking references to Java objects in a Java programming environment, said method comprising:

determining whether said Java command is likely to place the only reference to a Java object on the execution stack;

translating said command into another command when said determining determines that said Java command is likely to place the only reference to a Java object on the execution stack;

executing said Java command;

placing a reference to said object on a reference stack associated with said execution stack when said another command is executed; and

wherein said determining of whether said Java command is likely to place the only reference to a Java object on the execution stack is performed during Java Bytecode verification.

10. (Original) A method as recited in claim 9, wherein said determining that said Java command is likely to place the only reference to a Java object on the execution stack further comprises:

determining whether a Getfield, Aload, Getstatic, or Areturn command is being performed.

11. (Previously Presented) A method as recited in claim 10, wherein said determining that said Java command is likely to place the only reference to a Java object on the execution stack further comprises:

determining whether there is a change in the flow control.

12. (Original) A method as recited in claim 11, wherein said determining of whether said Java command is likely to place the only reference to a Java object on the execution stack further comprises:

determining whether a Putfield command is likely to overwrite a reference to an object on the execution stack before said reference is used.

13. (Original) A method as recited in claim 12, wherein said reference stack and said execution stack have the same size.

14. (Original) A method as recited in claim 13, wherein at least one reference to a Java object is stored in an entry with an offset that is the same as the offset used to store said at least one reference in said execution stack, when said another command is executed.

15. (Original) A Java Bytecode verifier suitable for operating in a Java operating environment,

wherein said Bytecode verifier operates to determine whether there is at least one Java command in a stream of Java Bytecode commands such that said at least one Java command is likely to place the only reference to a Java object on the execution stack;

wherein said Bytecode verifier operates to translate said Java command into another Java command when said Java command is likely to place the only reference to a Java object on the execution stack; and

wherein a reference associated with said command is placed on a reference stack as well as said execution stack when said another command is executed.

16. (Original) A Java Bytecode verifier as recited in claim 15, wherein said Bytecode verifier operates to determine whether a Getfield, Aload, Getstatic, or Areturn command is being performed.

17. (Original) A Java Bytecode verifier as recited in claim 16, wherein said Bytecode verifier operates to determine whether there is a change in the flow control.

18. (Original) A Java Bytecode verifier as recited in claim 16, wherein said Bytecode verifier operates to determine whether a Putfield command is likely to overwrite a reference to an object on the execution stack before said reference is used.

19. (Original) A Java Bytecode verifier as recited in claim 16, wherein said Bytecode verifier operates to:

determine whether there is a change in the flow control; and

determine whether a Putfield command is likely to overwrite a reference to an object on the execution stack before said reference is used.

20. (Previously Presented) A computer readable medium including computer program code for tracking references to objects of an object-oriented programming environment, said computer readable medium comprising:

computer program code for determining whether a command is likely to place a reference to an object on an execution stack of said object-oriented programming environment;

computer program code for determining whether there is a change in the flow control when said determining determines that said command is likely to place a reference to an object on an execution stack;

computer program code for translating said command into another command when said determining determines that there is a change in the flow control; and

computer program code for placing a reference to said object on a reference stack associated with said execution stack when said another command is executed.

21. (Previously Presented) A computer readable medium as recited in claim 20, wherein said object-oriented programming environment is a Java compliant operating environment.

(c) *Agesen et al.* does NOT teach or suggest determining whether there is a change in flow control in connection with determining whether a command is likely to place a reference to an object on an execution stack (Claim 1)

Agesen et al. et al. pertains to removal of reference conflicts (*Agesen et al.*, Title). It is respectfully submitted that *Agesen et al.* does NOT teach or suggest: determining control paths for the purpose or even in connection with determining: when an instruction is likely to place a reference on an execution stack. Accordingly, it is respectfully submitted that *Agesen et al.* cannot possibly teach or suggest this feature and claim 1 is therefore believed to be patentable over *Agesen et al.* for this additional reason.

(d) The Examiner has NOT made an prima facie case of obviousness because the Examiner has failed to provide a motivation or suggestion for combining *Agesen et al.* and *Steele Jr.*

It is very respectfully submitted that general allegation that *Agesen et al* and *Steele Jr.* can be combined to improve garbage collection is NOT enough to establish a prima facie case of obviousness (see, for example, MPEP §2143.01, paragraphs 1 and 3). The Examiner needs to provide a motivation or suggestion in the references themselves, or in the general art, for combining the references in the first place. In this case, the Examiner has failed to provide a motivation or suggestion for combining *Agesen et al.* and *Steele Jr.* as the Examiner has merely made a general allegation that the combination would improve garbage collection.

Moreover, in this case, there is no motivation to combine the reference because, among other things, *Agesen et al.* pertains to "removal of reference conflicts" and not determining whether a command is likely to place a reference to an object on an execution stack. Again, eliminating conflicts associated with control paths is NOT the same goal or motivation for determining whether a command is likely to place a reference on an execution stack. As such, there is NO need or motivation to combine the analysis of control path in *Agesen et al.* with the teaching of *Steele Jr.*